Resource-aware Broadcast Encryption for Selective-Sharing in Mobile Social Sensing

Akshay Dua
Portland State University
akshay@cs.pdx.edu

Nirupama Bulusu
Portland State University
nbulusu@cs.pdx.edu

***Abstract*—The rapid proliferation of smartphones has led to the emergence of mobile social sensing applications, spanning sharing of health data, location-based encounters, and transportation. A major concern for such applications is selective sharing, i.e., how does a user publish a sensor data stream confidentially to only authorized members in his/her social network? The needs of mobile sensing applications, such as dynamic communities and data dissemination from resource-constrained handhelds, make this problem more challenging than apparent. The novelty of this paper lies in the use of a cryptographic scheme called broadcast encryption to enable selective sharing for mobile social sensing. This is in contrast to unicast or pairwise encryption that is commonly used today. We evaluate state-of-art broadcast encryption schemes and note that they provide either efficiency, or adaptation to dynamic group sizes, but not both. We propose ECS (Extended Complete Subtree), a resource-aware broadcast encryption scheme that can efficiently support dynamic groups. We implement each encryption scheme on the Nokia N800 handheld device and demonstrate that ECS is more feasible than other schemes in terms of key storage, code size, and encryption and decryption efficiency.**

I. Introduction

The most transformational computing and communications revolution in the last decade has been the rapid proliferation of mobile handhelds and smart phones. Several interesting applications have become possible using data gathered from sensor and positioning-equipped mobile phones. These include the emergence of Mobile Health, Intelligent Transportation, and Mobile Social Networks. At UCLA, the DietSense project allows people to contribute pictures of their dietary intake using mobile camera phones and share it with their care providers [16]. Cartel [10] and BikeNet [6] enable drivers to share location, timestamps, and other information collected while driving or biking respectively. Social sensing applications like CenceMe [14] and PetrolWatch [5] enable sensory information sharing with friends to enable opportunistic location-based meetings, and to collect fuel price data respectively.

Unfortunately, the shared information may be confidential and *those who share the data may only want it accessible to a select few*. For example, CenceMe constantly infers user activities (e.g dancing), social contexts (e.g. party), and locations. A user may be comfortable with sharing the inferred information with close friends and family, but not co-workers, all of whom are also on her list of friends.

One approach to selective sharing would be to employ unicast or pairwise encryption schemes like AES (Advanced Encryption Standard) to exchange messages between devices. In unicast schemes, encrypting data for n users requires n encryptions. This quickly becomes inefficient in situations where, say, applications need to encrypt bandwidth-intensive video streams. Traditionally however, a more efficient cryptographic solution to the problem of selective sharing has been to use *broadcast encryption* [15], [4], [9].

In contrast to unicast schemes, broadcast encryption schemes enable a sender to encrypt data using $k \ll n$ encryptions. Furthermore, broadcast encryption schemes provide the ability to efficiently revoke users from a transmission. Thus, they adapt well to encrypting data for dynamic groups of users. For example, the CenceMe user, could use broadcast encryption to efficiently encrypt sensitive information to any subset of her friends.

In this paper, we evaluate the feasibility of implementing broadcast encryption on mobile devices. More specifically, our contributions are:

- Implementation and evaluation of several existing broadcast encryption schemes on the Nokia N800 devices.
- ECS (Extended Complete Subtree): a broadcast encryption scheme for selective information sharing using mobile devices that is efficient enough to be viable on mobile devices, and is scalable to typical social graph sizes [1].
- We find that unlike many algorithms that must be setup with a maximum number of friends, ECS does not. Thus, ECS can be scaled gradually. Furthermore, when compared to algorithms that don't need a pre-determined maximum, ECS has lower encryption and decryption overhead.

II. System Model

In this section, we describe our system model. Our goal is to make broadcast encryption practical on mobile platforms. We assume that a sender, Alice, uses her mobile platform to send a secret message to a subset of her friends. Even if the message were to fall into the hands of a friend not in the subset, its contents should be inaccessible. The encryption scheme should allow the subset of friends to be chosen on the fly. Figure 1 shows a sender sharing a secret message M with her family. She encrypts M specifically for her family members and sends it to a web portal that forwards it to each of them. The web portal is not a necessary component of the model, but does make the communication more efficient.

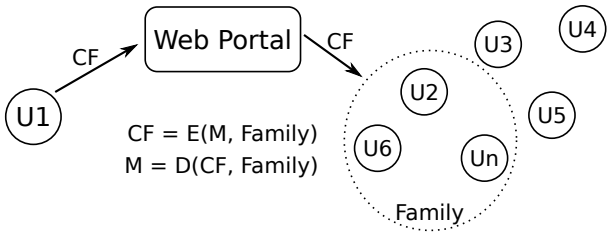


Fig. 1. System Model: sender $U1$ has friends $\{U2, U3, \ldots, Un\}$ and would like to send a secret message to a subset of them.

III. Background

In this section, we describe the concepts underpinning broadcast encryption schemes. Several of these schemes were designed to support access control on streaming paid television content. The schemes assume a single broadcaster with abundant computational resources and potentially millions of receivers. However, the resource assumptions are not practical for a sender using a mobile device [4], [15]. Our challenge is to build a broadcast encryption scheme for efficient selective sharing on mobiles.

A *Broadcast Encryption* scheme enables a sender to encrypt a message for any arbitrary subset S of receivers $1, ..., N$ who are listening on a broadcast channel [3], [7]. Any receiver in S can decrypt the broadcast using their private key. Usually, the broadcast-encrypted content is a session key rather than the data in question. The session key can then be used to encrypt the data with an algorithm of choice. Thus, for the purposes of this work, *ciphertext* in this paper refers to an encrypted session key. Furthermore, the act of changing the aforementioned session key is called *re-keying*.

In the following sections, we consider three well known broadcast encryption schemes: the pairing-based scheme developed by Boneh and Waters [4], the subset cover system by Naor et al [15], and the Logical Key Hierarchy scheme (LKH) in "batch" mode as described in [12] by Li et al. We discuss each scheme briefly in the following sections.

A. Boneh-Waters

Boneh-Waters is a public key cryptosystem that provides the following functions.

$Setup(N, \ldots)$ generates the public and private keys needed by the scheme. It outputs a public key PK and private keys $SK_1, \ldots, SK_n$, where SK_u is given to user u. The input N is the number of users the system should be setup with. Note that the number of users must be determined beforehand.

Name	Number of Encryptions	Receiver Storage	Sender Storage	Number of Decryptions
Boneh	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	1
Comp. Subtree	$rlog(N/r)$	$logN$	$2N-1$	1
Subset Diff	$2r-1$	$1/2log^2N$	$2N-1$	1
LKH	$rlog(n/r)$	$log(n)$	$2n-1$	1

TABLE I
Algorithmic complexity of encryption schemes

$Encrypt(S, PK, M, \ldots)$ is the encryption algorithm wherein PK is the public key, S the subset of users, and $M = K_s$ is the session key. It outputs a ciphertext C that can be successfully decrypted by a private key belonging to any user in S.

$Decrypt(SK_j, C, \ldots)$ is used to decrypt a ciphertext C. The output is the message $M = K_s$ as long as M was intended for user with private key SK_j. For more details please see [2].

B. Subset Cover

The subset cover group of algorithms consists of the *complete subtree* scheme and the *subset difference* scheme. Both are based on symmetric cryptography but provide a similar interface as Boneh, namely:

$$(Setup(N), Encrypt(S, M), Decrypt(j, C))$$

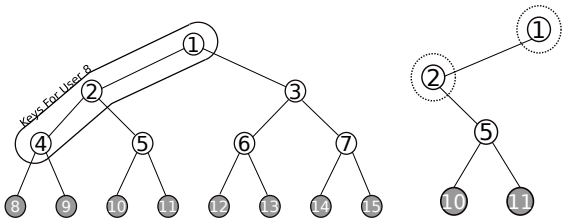
These schemes are, (i) *stateless*, because users need not update their keys across sessions; once the keys are obtained, they are valid for the life of the system, and (ii) *static*, because the schemes are initialized with the maximum number of receivers that will ever use the scheme. It should be noted that by this definition, the Boneh scheme would also be considered a stateless and static scheme.

1) Complete Subtree: The complete subtree method is based on a key tree with N leaves, one for each user (see Figure 2(a)). Each user gets all $log(N)$ keys along the path to the root. So for example, user 8 gets the keys $\{K_8, K_4, K_2, K_1\}$. If no users are revoked from the transmission, the session key is distributed by broadcasting the message $E_{K_1}(K_s)$ to all the users. Here, E is a symmetric encryption algorithm (like DES), K_1 is the key used for encryption and K_s is the group session key. Since, each user has K_1 they can derive K_s using $D_{K_1}(K_s)$ where D is the corresponding decryption algorithm. Notice that K_s will be used to encrypt the actual data that is transmitted. The algorithm used for that purpose can be E.

Revoking users involves constructing a minimum spanning tree from the root to the revoked users, called a *cover*. As an example, consider that users 10 and 11 are being revoked, the resulting minimum spanning tree is shown in Figure 2(b). Here, the nodes with out-degree 1 ($\{1, 2\}$) in the cover are particularly interesting because the keys associated with their missing children will be used to encrypt the session key K_s. So, the message that will be broadcast is $\{E_{K_4}(K_s), E_{K_3}(K_s)\}$. Note that revoked users cannot obtain the session key since none of them have keys K_4 or K_3 required to successfully decrypt the message.

2) Subset Difference: The subset difference algorithm is more involved. A subset of users $S_{i,j}$ is defined as $S_i \setminus S_j$ where S_i (or S_j) is the set of leaves below node i (or j). Each such subset is associated with a label $Lab_{i,j}$ and a key $K_{i,j}$. A user is given the labels to all subsets $S_{i,j}$ where i is a node on its path and j is a node *hanging off* its path to the root. Thus node 8 (in Figure 2(a)) would get the labels:

$$\{Lab_{1,3}, Lab_{1,5}, Lab_{1,9}\}$$
$$\{Lab_{2,5}, Lab_{2,9}\}$$
$$\{Lab_{4,9}\}$$



(a) Key tree used in the complete subtree scheme (b) Revoking 10 & 11

Fig. 2. The complete subtree scheme

These labels can then be used to *derive* keys for *any* subset the user belongs to, even those it does not have labels for. This due to the way labels were created to begin with. For details on the algorithm please refer to the paper by Naor et al. [15]. The subset difference scheme was primarily designed to reduce the generated cipher text size when revoking users. However, the trade-off was an added storage cost at the receiver.

C. LKH

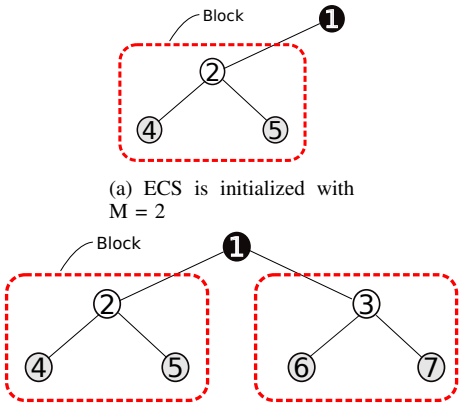
LKH is very similar to the complete subtree algorithm, but with two key differences. (i) It is *stateful*, i.e., the keys for the system change when membership changes. If users miss key updates, they need to contact the server to obtain the current ones. (ii) It is *dynamic* because unlike the subset cover scheme it does not need to know the maximum number of users N before-hand. LKH adds or removes users to the tree when they join or leave.

In LKH "batch" mode, users can be removed a batch at a time rather than individually as prescribed by the original LKH scheme [17], [18]. Revoking users thus becomes very similar to revoking users in the complete subtree algorithm, with the difference being that keys pertaining to nodes in the tree may change. Further, since users (i.e. leaves of the tree) will be joining and leaving, the height and the intermediate nodes in the tree may change as well. Another subtle difference, is that adding users in LKH requires an update to existing users while this is not the case in other schemes.

Table I presents the algorithmic complexity of each scheme. One might be inclined to pick a suitable system based on the summary in Table I, however, note that the expressions represent worst case scenarios and that key sizes, real encryption times, and communication overhead are not clear from complexity expressions alone.

IV. ECS: Resource-Aware Broadcast Encryption

The advantage with *stateless* schemes is that users need not be online to decrypt received messages. This is largely due to the fact that keys belonging to the scheme do not change once they are created. However, the disadvantage is that the schemes are *static* — the maximum number of users N_{max} that might ever use the system needs to be known before hand. If such information is not available, then the system will need to be setup with some large estimate of N_{max} and this will result in the client and server storing a larger number of keys and requiring more computation when re-keying.



(a) ECS is initialized with M = 2

(b) Third user arrives and ECS is scaled

Fig. 3. The ECS scheme is initialized with a block size of 2 and then scaled when the third user arrives

The advantage with a scheme like LKH is that the key tree is only ever big enough to support the number of users actively using the scheme (i.e. the scheme is *dynamic*). However, the disadvantage is that clients need to be online all the time to not miss any key updates (i.e. the scheme is *stateful*). Considering that our system model (Figure 1) assumes mobile users in a cellular network, a user's mobile phone will be prevented from sleeping and key updates will be missed in places without coverage, or when the phone batteries are dead.

A good compromise would be a *stateless* scheme that could *dynamically* grow to accommodate new users. Of course, it is always possible to just re-initialize any of the stateless schemes with a larger N_{max}, but then this would require the entire group of users to be re-keyed with new keys individually through secure unicast channels like SSL. Further, this does not solve the problem of estimating a good value for N_{max}.

We present *Extended Complete Subtree (ECS)*, a new broadcast encryption scheme that has a combination of both the properties desirable on a mobile platform: being *simultaneously* stateless and dynamic. ECS combines multiple instantiations of the complete subtree scheme into a single broadcast encryption scheme. More specifically, ECS consists of a root node with multiple subtrees, each of which operate under the complete subtree scheme.

ECS is first initialized with M users where M is a power of two. We refer to M as the *block size* and the system is then scaled in multiples of these block sizes. Figure 3(a) shows the ECS scheme initialized with $M = 2$. Here, the subtree rooted at node 2 is called a *block* and node 1 is the new root. In this example, client 4 gets keys $\{1, 2, 4\}$ while client 5 gets keys $\{1, 2, 5\}$. When the third user arrives, the system is scaled by adding a new subtree with M leaves i.e. a new block, to the root (see figure 3(b)). Now, the new clients (represented by nodes 6 and 7 in Figure 3(b)) get the keys $\{1, 3, 6\}$ and $\{1, 3, 7\}$ respectively. Notice that since the key associated with the root node has not changed, no key updates need to be sent to the existing users. As more users arrive, more blocks can be added to the root node. The cover finding algorithm remains

essentially the same as described in section III-B1 except that the cover is now an aggregate of all the block covers.

Another advantage of this scheme, is that the key storage requirements at each receiver remain constant at $log(M)$ regardless of the number of users joining the group. For a given maximum number of users, N_{max}, the sender and receiver-side key storage are proportional to the choice of M. However, the relationship between M and ciphertext size is not that simple. When M is at its smallest i.e. $M = 2$, the ciphertext size is $O(N_{max})$. When $M \approx N_{max}$, then ciphertext size is similar to the complete subtree scheme at $O(log(N_{max}))$.

Figure 4 shows the change in ciphertext size with block size for a set of maximum number of users $N_{max} = \{128, 256, 512\}$, and block sizes that ranged from $32 \leq M \leq N_{max}$. The ciphertext size was computed after randomly revoking 20% of the users and repeating the test 40 times for each block size. For each value of N_{max}, we can see that ciphertext size is minimum when block size $M = 64$.

V. Results

This section compares ECS with the broadcast encryption schemes described in Section III. We show how ECS is both *stateless* and *dynamic*, thus making it more suitable to use on mobile platforms. Our goal is to determine whether ECS matches the efficiency of static schemes while supporting dynamic user groups.

A. Experimental Setup

All broadcast encryption schemes had to be implemented from scratch since no readily available implementations were found. The Boneh-Waters scheme was implemented using the Pairing Based Crypto (PBC) library developed at Stanford University by Benn Lynn [13]; LKH was implemented as described by Li et al. [12]; and the subset cover group of algorithms were implemented as described by Naor et al. [15]. All schemes except Boneh-Waters use symmetric encryption and for that we used the DES algorithm with 256 bit keys.

Senders, receivers and each of the broadcast encryption schemes were implemented in C and deployed on Nokia N800 devices. In each of the experiments, a sender is assumed to have a maximum of $N_{max} \in \{512, 1024\}$ friends. A fixed N_{max} is needed for broadcast encryption schemes that require it as a setup parameter. At any given time, a sender may have $N_{active} \leq N_{max}$ friends. Additionally, ECS is setup with a block size $M = 64$ (see Section IV).

To confidentially broadcast a message, a sender first encrypts it for a desired subset of $1 \leq n \leq N_{active}$ friends using one of the aforementioned encryption schemes and sends it to a web server. The web server then forwards it to the respective set of n friends.

B. Key Storage

We start by comparing sender and receiver-side key storage requirements for each of the broadcast encryption schemes. Figure 5 compares sender-side key storage requirements. Recall that Boneh-Waters has a single public key where as the rest have multiple smaller keys (256 bits each). Also, the number of such keys is a function of N_{max} for the subset cover group of schemes and Boneh-Waters, whereas they are a function of N_{active} for LKH and ECS. We can see that LKH and ECS requirements grow linearly with the number of active users. The subset cover group of schemes on the other hand, are the most expensive and the storage cost remains constant throughout. Unicast schemes have the minimum storage cost since the sender needs to store only one symmetric key per receiver.

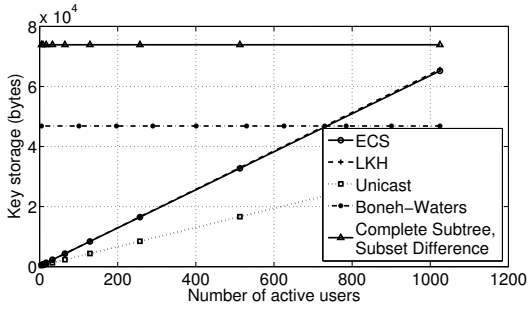


Fig. 5. Sender-side key storage

Figure 6 shows receiver-side key storage requirements. We can see that among all broadcast encryption schemes, ECS has the minimum storage requirements that remain constant regardless of N_{active}. This is because ECS's fixed block size. Unicast encryption requires the least amount of storage since each receiver only needs to store a single 256 bit key.

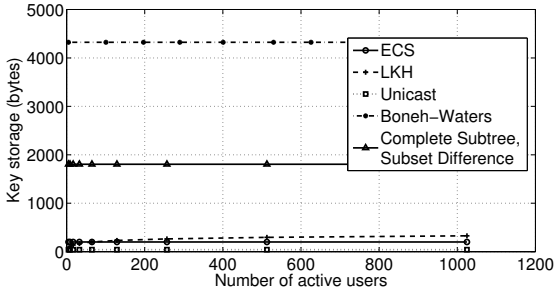
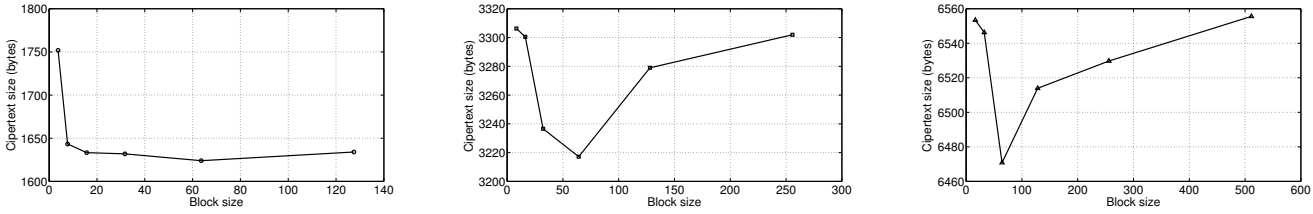


Fig. 6. Receiver-side key storage

C. Encryption Efficiency

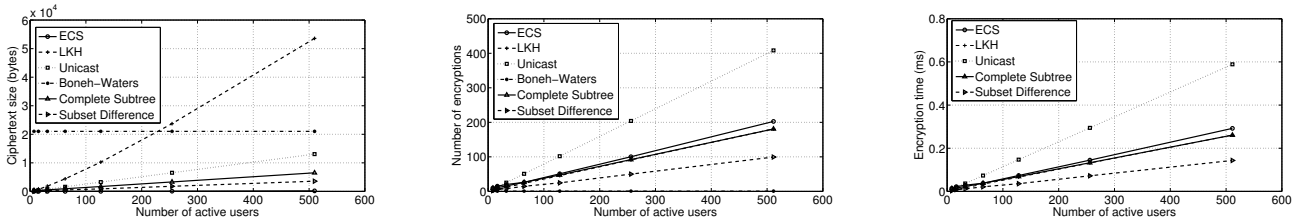
We evaluate the cost of creating a ciphertext message. Recall that the message being encrypted is a 256-bit session key. We compare ciphertext size (Figure 7(a)), number of encryptions required to create the ciphertext (Figure 7(b)), and the time required for its creation (Figure 7(c)). For each of the N_{active} receivers, the ciphertext was created for a random subset of 80% of them. The experiment was then repeated 40 times for each value of N_{active}.

Notice that LKH quickly becomes more expensive than all the other algorithms. The reason for this, is that unlike other



(a) Number of users n = 128 (b) Number of users n = 256 (c) Number of users n = 512

Fig. 4. Finding a suitable block size (M) for the ECS scheme



(a) Ciphertext message size (b) Number of encryptions (c) Encryption time

Fig. 7. Comparing the cost of creating a ciphertext

algorithms which only need to encrypt a new session key and broadcast it to the group, LKH actually changes many of the keys in its tree. These keys are then encrypted and sent to the respective groups of receivers.

In contrast to other schemes, Boneh-Waters has a constant size ciphertext (dependent only on N_{max}) regardless of the number of active users. However, the ciphertext is still substantially larger than the subset cover schemes and ECS. Also, notice that the ciphertext size for ECS is similar to that of complete subtree. Thus, we have not adversely impacted the ciphertext size, but have still gained the ability to scale the system dynamically.

Figure 7(b) shows that LKH, ECS and the complete subtree scheme have similar number of encryptions. Boneh-Waters requires a single encryption regardless of the number of active users. However, the time required by that encryption is prohibitive — ≈ 33 secs) — which is why the plot for Boneh-Waters has not been included in Figure 7(c). Finally, we can see that for large number of active users, unicast encryption is most expensive.

D. Decryption Efficiency

Another important energy and efficiency consideration is the cost of decrypting the ciphertext. We evaluate this cost based on amount of time taken for decryption as well as the energy consumed. Figure 8 shows the decryption time using each scheme for various values of N_{active}. The decryption time for all symmetric broadcast encryption schemes is the same since only a single decryption is required per ciphertext. The same is true for Boneh-Waters, however the decryption time is quite large, approximately 10 secs. This is mainly due to the resource-intensive mathematical operations required to perform the decryption.

E. Energy Consumption

Table II shows the mean energy consumed (in Joules) by the various algorithms while decrypting a ciphertext message. Again, the symmetric broadcast encryption schemes and the unicast scheme consume similar amounts of energy, while Boneh-Waters was significantly more expensive.

Complete Subtree, ECS, LKH, Unicast, Subset Difference	Boneh
2.55×10^{-4} J	6.63 J

TABLE II
RECEIVER-SIDE ENERGY CONSUMPTION IN JOULES

F. Code Size

It is important that resource requirements at the receiver remain practical for personal devices like cell phones and PDAs. Table III shows the code sizes for the various schemes that we implemented. The code size of the receiver-side program, excluding any encryption library, is also shown. The complete subtree scheme is the smallest mainly due to the simplicity of its implementation on the client side. Notice

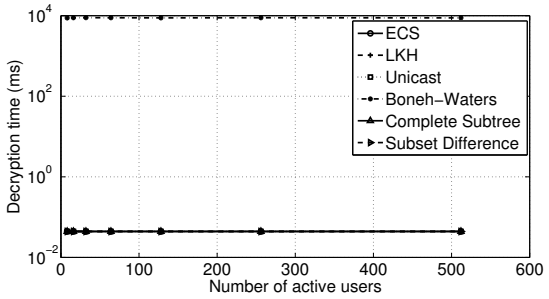


Fig. 8. Time taken to decrypt ciphertext

Broadcast Encryption Library (Kbytes)					Other Code (Kbytes)
Complete Subtree	ECS	Subset Difference	LKH	Boneh	Producer/Consumer
0.42	0.47	4.75	1.93	1.28	3

TABLE III
CLIENT SIDE CODE SIZE FOR VARIOUS BTR LIBRARIES AS WELL AS THE PRODUCER/CONSUMER API

that ECS is very similar to the requirements by complete subtree since the decryption algorithm remains unchanged. Although the size of the Boneh library as implemented is 1.28 Kbytes, its dependency on the PBC library (175 Kbytes) [13] and the GNU Multi-Precision Library (150 Kbytes) [8] effectively makes it much larger. The size of the rest of the code for communicating with the web portal and providing a common interface to the encryption libraries is shown in the last column.

VI. RELATED WORK

Significantly, no prior work has focused on efficient and scalable selective sharing for mobile social networks. Zhu et al[19] propose a group re-keying scheme for ad-hoc sensor networks that exploits the property of each member being a host and a router to create secure hop-by-hop channels to re-key the group. This is quite different from our scenario where a sender may encrypt data for any subset of users. Lazos et al [11] adopted the LKH scheme also for ad-hoc networks. However, the resource and communication overhead incurred is of the same order as the original LKH scheme [17], [18]. As we have shown, the LKH scheme is quite expensive for groups with even few hundred members.

VII. CONCLUSION

Emerging social sensing applications like health data sharing, location-based encounters, and transportation need an efficient, scalable, and secure method for selectively sharing sensitive information. A broadcast encryption scheme the sender to simultaneously encrypt data for a given set of receivers. Traditional broadcast encryption schemes are either computationally expensive or do not adapt gracefully to dynamic group sizes. We addressed this problem by proposing a new resource-aware broadcast encryption scheme called ECS (Extended Complete Subtree). We have shown that ECS can be implemented very efficiently for cell phone class devices, that it scales to thousands of users per group, naturally supports large and dynamic groups, and is power efficient.

REFERENCES

[1] Anatomy of Facebook. https://www.facebook.com/notes/facebook-data-team/anatomy-of-facebook/10150388519243859.

[2] P.S.L.M. Barreto, B. Lynn, and M. Scott. Efficient Implementation of Pairing-Based Cryptosystems. *Journal of Cryptology*, 17(4):321–334, 2004.

[3] D. Boneh, C. Gentry, and B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. *LECTURE NOTES IN COMPUTER SCIENCE*, 3621:258, 2005.

[4] D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 211–220. ACM Press New York, NY, USA, 2006.

[5] Y. F. Dong, Salil S. Kanhere, Chun Tung Chou, and Nirupama Bulusu. Automatic collection of fuel prices from a network of mobile cameras. In Sotiris E. Nikoletseas, Bogdan S. Chlebus, David B. Johnson, and Bhaskar Krishnamachari, editors, *DCOSS*, volume 5067 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2008.

[6] SB Eisenman, E. Miluzzo, ND Lane, RA Peterson, GS Ahn, and AT Campbell. The BikeNet mobile sensing system for cyclist experience mapping. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 87–101. ACM New York, NY, USA, 2007.

[7] A. Fiat and M. Naor. Broadcast encryption. *Proceedings of the 13th annual international cryptology conference on Advances in cryptology table of contents*, pages 480–491, 1994.

[8] T. Granlund. The GNU MP LIBRARY, version 2.0. 2, June 1996. *Online: http://gmplib.org/*.

[9] H. Harney and E. Harder. Logical Key Hierarchy Protocol. *Intemet Draft, draft-harney-sparta-lkhp-sec-00. txt, Internet Engineering Task-Force*, 3, 1999.

[10] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138. ACM New York, NY, USA, 2006.

[11] L. Lazos and R. Poovendran. Energy-aware secure multicast communication in ad-hoc networks using geographic location information. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 4, 2003.

[12] X.S. Li, Y.R. Yang, M.G. Gouda, and S.S. Lam. Batch rekeying for secure group communications. In *Proceedings of the 10th international conference on World Wide Web*, pages 525–534. ACM New York, NY, USA, 2001.

[13] B. Lynn. PBC library. *Online: http://crypto.stanford.edu/pbc/*.

[14] E. Miluzzo, N.D. Lane, S.B. Eisenman, and A.T. Campbell. CenceMe-Injecting Sensing Presence into Social Networking Applications. *LECTURE NOTES IN COMPUTER SCIENCE*, 4793:1, 2007.

[15] D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 41–62, 2001.

[16] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen. Image browsing, processing, and clustering for participatory sensing: lessons from a DietSense prototype. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 13–17. ACM Press New York, NY, USA, 2007.

[17] D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. *Work in Progress*.

[18] C.K. Wong, M. Gouda, and SS Lam. Secure group communications using key graphs. *Networking, IEEE/ACM Transactions on*, 8(1):16–30, 2000.

[19] S. Zhu, S. Setia, S. Xu, and S. Jajodia. GKMPAN: An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks. *Journal of Computer Security*, 14(4):301–325, 2006.